	I	**I**gnore the special lookup program.
	K	Primary **K**ey is used as starting index for the lookup.
	L	**L**earning a new entry is allowed.
	M	**M**ultiple-index lookup allowed.
	N	Internal **N**umber lookup allowed (but not forced).
	O	**O**nly find one entry if it matches exactly.
	Q	**Q**uestion erroneous input (with two ??).
	S	**S**uppresses display of .01 (except B cross-reference match) and of any Primary Key fields.
	T	Con**T**inue searching all indexes until user selects an entry or enters ^^ to get out.
	U	**U**ntransformed lookup.
	V	**V**erify that looked-up entry is OK.
	X	E**X**act match required.
	Z	**Z**ero node returned in Y(0) and external form in Y(0,0).

X If DIC(0) does not contain an A, then the variable X must be defined equal to the value you want to lookup. If the value in X has more than one match or partial match, the lookup fails and Y=-1 is returned.

If the lookup index is compound (i.e., has more than one data subscript), then X can be an array X(n) where "n" represents the position in the subscript. For example, if X(2) is defined, it will be used as the lookup value to match to the entries in the second subscript of the index. If only the lookup value X is passed, it will be assumed to be the lookup value for the first subscript in the index, X(1).

DIC("A") (Optional) A prompt that is displayed prior to the reading of the X input. If DIC("A") is not defined, the word Select, the name of the file, [i.e., $P(^GLOBAL(0),"^",1)], a space, the LABEL of the .01 field, and a colon will be displayed. If the file name is the same as the LABEL of the .01 field, then only the file name will be displayed. DIC(0) must contain an A for this prompt to be issued. For example, if the EMPLOYEE file had a .01 field with the LABEL of NAME, then FileMan would issue the following prompt:

```
Select EMPLOYEE NAME:
```

By setting DIC("A")="Enter Employee to edit: ", the prompt would be:

```
Enter Employee to edit:
```

Notice that it is necessary for the prompt in DIC("A") to include the colon and space at the end of the prompt if you want those to be displayed.

If the lookup index is compound (i.e., has more than one data subscript), then DIC("A") can be an array DIC("A",n) where "n" represents the position in the subscript. For example, DIC("A",2) will be used as the prompt for the second subscript in the index. If only the single prompt DIC("A") is passed, it will be assumed to be the prompt for the first subscript in the index DIC("A",1).

If DIC("A",n) is undefined for the 'nth' subscript, then the 'Lookup Prompt' field for that subscript from the INDEX file will be used as the prompt, or if it is null, the LABEL of the field from the data dictionary.

DIC("B") (Optional) The default answer which is presented to the user when the lookup prompt is issued. If a terminal user simply presses the Enter/Return key, the DIC("B") default value will be used, and returned in X. DIC("B") will only be used if it is non-null.

If the lookup index is compound (i.e., has more than one data subscript), then DIC("B") can be an array DIC("B",n) where "n" represents the position in the subscript. For example, DIC("B",2) will be used as the default answer for the prompt for the second subscript in the index. If only the single default answer DIC("B") is passed, it will be assumed to be the default answer for the prompt for the first subscript in the index DIC("B",1).

DIC("DR") When calling DIC with LAYGO allowed, you can specify that a certain set of fields will be asked for in the case where the user enters a new entry. This list is specified by setting the variable DIC("DR") equal to a string that looks exactly like the DR string of fields that is specified when calling ^DIE. Such a list of what VA FileMan calls forced identifiers overrides any identifiers that would normally be requested for new entries in this file.

DIC("P") **NOTE:** As of Version 22 of FileMan, the developer is no longer required to set DIC("P"). The only exception to this is for a few files that are not structured like a normal FileMan file, where the first subscript of the data is variable in order to allow several different 'globals' to use the same DD. An example of this is the FileMan Audit files where the first subscript is the file number of the file being audited.

This variable is needed to successfully add the FIRST subentry to a multiple when the descriptor (or header) node of the multiple does not exist. In that situation, DIC("P") should be set equal to the subfile number and subfile specifier codes for the multiple. (See the File Header section of the Global File Structure chapter.) If the descriptor node for the multiple already exists, DIC("P") has no effect.

In order to automatically include any changes in the field's definition in DIC("P"), it is best to set this variable to the second ^-piece of the 0-node of the multiple field's definition in the DD. (See the Field Definition section of the Global File Structure chapter.)

Thus, for example, if file 16150 had a multiple field #9, set DIC("P") like this:

```
S DIC("P")=$P(^DD(16150,9,0),"^",2)
```

	For more information, see Adding New Subentries to a Multiple below.
DIC("PTRIX",f,p,t)=d	**DIC("PTRIX",f,p,t)=d** where **f** is the from (pointing) file number, **p** is the pointer field number, **t** is the pointed-to file number, and **d** is an "^" delimited list of index names. When doing a lookup using an index for a pointer or variable pointer field, this new array allows the user to pass a list of indexes that will be used when searching the pointed-to file for matches to the lookup value. For example, if your file (662001) has a pointer field (5) to file 200 (NEW PERSON), and you wanted the lookup on file 200 to be either by name ("B" index), or by the first letter of the last name concatenated with the last 4 digits of the social security number ("BS5" index): DIC("PTRIX",662001,5,200)="B^BS5". Note that if the call allows records to be added to a pointed-to file, then the list in the "PTRIX" entry should contain the "B" index. However, the "B" index would not need to be included in the list if the first index in the "PTRIX" array entry is a compound index whose first subscript is the .01 field.
DIC("S")	(Optional) DIC("S") is a string of M code that DIC executes to screen an entry from selection. DIC("S") must contain an IF statement to set the value of $T. Those entries that the IF sets as $T=0 will not be displayed or selectable. When the DIC("S") code is executed, the local variable Y is the internal number of the entry being screened and the M naked indicator is at the global level @(DIC_"Y,0)"). Therefore, to use the previous example again, if you wanted to find a male employee whose name begins with SMITH, you would:

```
S DIC="^EMP(",DIC(0)="QEZ",X="SMITH"
S DIC("S")="I $P(^(0),U,2)=""M"""
D ^DIC
```

DIC("T")	(Optional) Present every match to the lookup value, quitting only when user either selects one of the presented entries, enters ^^ to quit, or there are no more matching entries found. Currently, if one or more matches are found in the first pass through the indexes, then FileMan quits the search, whether or not one of the entries is selected. Only if no matches are found in the first pass does FileMan continue on to try transforms to the lookup value. This includes transforms to find internal values of pointers, variable pointers, dates or sets. Another feature of the “T” flag is that indexes are truly searched in the order requested. If, for example, an index on a pointer field comes before an index on a free-text field, matches from the pointer field will be presented to the user before matches to the free-text field. When used in combination with the “O” flag, all indexes will be searched for an exact match. Then, only if none are found, will FileMan make a second pass through the indexes looking for partial matches.
DIC("V")	If the .01 field is a variable pointer, it can point to entries in more than one file. You can restrict the user's ability to input entries from certain files by using the DIC("V") variable. It is used to screen files from the user. Set the DIC("V") variable to a line of M code that returns a truth value when executed. The code is executed after someone enters data into a variable pointer field. If the code tests false, the user's input is rejected; FileMan responds with ?? and a "beep." If the lookup index is compound (i.e., has more than one data subscript), and if any of the subscripts index variable pointer fields, then DIC("V",n) can be passed where "n" represents the subscript position of the variable pointer field in the index. For example, if DIC("V",2) is passed in, it will be used as the screen for

files pointed-to by the variable pointer field indexed in the second subscript of the index. If only the entry DIC("V") is passed, it will be assumed to be the variable pointer file screen for the first subscript in the index, DIC("V",1).

When the user enters a value at a variable pointer field's prompt, VA FileMan determines in which file that entry is found. The variable Y(0) is set equal to information for that file from the data dictionary definition of the variable pointer field. You can use Y(0) in the code set into the DIC("V") variable. Y(0) contains:

^-Piece	**Contents**
Piece 1	File number of the pointed-to file.
Piece 2	Message defined for the pointed-to file.
Piece 3	Order defined for the pointed-to file.
Piece 4	Prefix defined for the pointed-to file.
Piece 5	y/n indicating if a screen is set up for the pointed-to file.

All of this information was defined when that file was entered as one of the possibilities for the variable pointer field.

For example, suppose your .01 field is a variable pointer pointing to files 1000, 2000, and 3000. If you only want the user to be able to enter values from files 1000 or 3000, you could set up DIC("V") like this:

```
S DIC("V")="I +Y(0)=1000!(+Y(0)=3000)"
```

DIC("W") (Optional) An M command string which is executed when DIC displays each of the entries that match the user's input. The condition of the variable Y and of the naked indicator is the same as for DIC("S"). If DIC("W") is defined, it overrides the display of any identifiers of the file. Thus, if DIC("W")="", the display of identifiers will be suppressed.

NOTE: DIC("W") is killed by ^DIC calls.

DIC("?N",file#)=n	The number "n" should be an integer set to the number of entries to be displayed on the screen at one time when using "?" help in a lookup. Usually, file# will be the number of the file on which you're doing the lookup. However, if doing a lookup using an index on a pointer field, and if DIC(0) contains "L", then the user also is allowed to see a list of entries from the pointed-to file, so in that case file# could be the number of that pointed-to file. For example, when doing a lookup in test file 662001, if the developer wants only five entries at a time to be displayed in question-mark help, set DIC("?N",662001)=5
DLAYGO	(Optional) If this variable is set equal to the file number, then the users will be able to add a new entry to the file whether or not they have LAYGO access to the file. This variable, however, does not override the checks in the LAYGO nodes of the data dictionary. Those checks must still prove true for an entry to be added. **NOTE:** In addition, DIC(0) must contain L to allow addition of entries to the file.

Output Variables

Y

DIC always returns the variable Y. The variable Y is returned with one of these three formats:

Y=-1	The lookup was unsuccessful.
Y=N^S	N is the internal number of the entry in the file and S is the value of the .01 field for that entry.
Y=N^S^1	N and S are defined as above and the 1 indicates that this entry has just been added to the file.

Y(0)

This variable is only set if DIC(0) contains a Z. When the variable is set, it is equal to the entire zero node of the entry that was selected.

Y(0,0)

This variable also is only set if DIC(0) contains a Z. When the variable is set, it is equal to the external form of the .01 field of the entry.

The following are examples of returned Y variables based on a call to the EMPLOYEE file stored in ^EMP(:

```
S DIC="^EMP(",DIC(0)="QEZ",X="SMITH"
D ^DIC
```

Returned are:

```
Y      = "7^SMITH,SAM"
Y(0)   = "SMITH,SAM^M^2231109^2
Y(0,0) = "SMITH,SAM"
```

If the lookup had been done on a file whose .01 field points to the EMPLOYEE file, the returned variables might look like this:

```
Y      = "32^7"  [ Entry #32 in this file and #7
                 in EMPLOYEE file.]
Y(0)   = "7^RX 2354^ON HOLD"
Y(0,0) = "SMITH,SAM"  [.01 field of entry 7 in
                       EMPLOYEE file]
```

X

Contains the value of the field where the match occurred.

If the lookup index is compound (i.e., has more than one data subscript), and if DIC(0) contains "A" so that the user is prompted for lookup values, then X will be output as an array X(n) where "n" represents the position in the subscript and will contain the values from the index on which the entry was found. Thus, X(2) would contain the value of the second subscript in the index. If possible, the entries will be output in their external format (i.e., if the subscript is not computed and doesn’t have a transform). If the entry is not found on an index (example, when lookup is done with X=" " (the space-bar return feature)), then X and X(1) will contain the user input, but the rest of the X array will be undefined.

DTOUT

This is only defined if DIC has timed-out waiting for input from the user.

DUOUT	This is only defined if the user entered an up-arrow.

DIC(0) Input Variables in Detail

The effects of the various characters which can be contained in DIC(0) are described below:

A DIC asks for input from the terminal and asks again if the input is erroneous. A response of null or a string containing ^ is accepted. Input is returned in X when DIC quits. If DIC(0) does not contain the character A, the input to DIC is assumed to be in the local variable X.

B Without the B flag, if there are cross-referenced pointer or variable pointer fields in the list of indexes to use for lookup and if DIC(0) contains "M" and there is no screening logic on the pointer that controls the lookup on the pointed-to file, then:

1. For each cross-referenced pointer field, FileMan checks ALL lookup indexes in each pointed-to file for a match to X (time-consuming);

2. If X matches any value in any lookup index (not just the "B" index) on the pointed-to file and the IEN of the matched entry is in the home file's pointer field cross-reference, FileMan considers this a match. This may perhaps not be the lookup behavior you wanted, see Examples section.

The B flag prevents this behavior by looking for a match to X only in the B index (.01 field) of files pointed to by cross-referenced pointer or variable pointer fields. This makes lookups quicker and avoids the risk of FileMan matching an entry in the pointed-to file based on some unexpected indexed field in that file.

C Normally, when DIC does a lookup and finds an entry that matches the input, that entry is presented to the user only once even if the entry appears in more than one cross-reference. This is called cross-reference suppression and can be overridden by including a C in DIC(0). If, for example, a person with the name ZACHARY,DAVID is an entry in a file, then his name will appear in the B cross-reference of the file. If he has a nickname of ZACH, which is in the C cross-reference of the file, then when a user enters ZACH as a lookup value, the name, ZACHARY,DAVID, will appear only once in the choices. But if there is a C in DIC(0), then ZACHARY,DAVID will appear twice in the choices; once as a hit in the B cross-reference and again as a hit in the C cross-reference.

F Prevents saving the entry number of the matched entry in the ^DISV global. Ordinarily, the entry number is saved at ^DISV(DUZ,DIC). This allows the user to do a subsequent lookup of the same entry simply by pressing the space bar and Enter/Return key. To avoid the time cost of setting this global, include an F in DIC(0).

I If DIC(0) contains I, any special user-written lookup program for a file will be ignored and DIC will proceed with its normal lookup process.

You can write a special lookup program to be used to find entries in a particular file. This special program can be defined by using the Edit File option of the Utility Functions submenu (see the Special Lookup Programs section in the Advanced File Definition chapter.) When a lookup program is defined, VA FileMan will bypass the normal lookup process of DIC and branch to the user written program. This user written lookup program must respond to the variables documented in this section and provide the functionality of DIC as they pertain to the file.

K This flag causes ^DIC to use the Uniqueness index for the Primary Key as the starting index for the lookup, rather than starting with the B index. (If developers want to specify some other index as the starting index, then they can specify the index by using the "D" input variable, and either the IX^DIC or the MIX^DIC1 call instead of ^DIC.)

L If DIC(0) contains L and the user’s input is in valid format for the file’s .01 field, then DIC will allow the user to add a new entry to the file at this point (Learn-As-You-GO), as long as at least one of these four security-check conditions is true:

The local variable DUZ(0) is equal to the @-sign.

If Kernel’s File Access Security System (formerly known as Kernel Part 3) is being used for security, the file is listed in the user’s record of accessible files with LAYGO access allowed.

If file access management is not being used, a character in DUZ(0) matches a character in the file's LAYGO access code or the file has no LAYGO access code.

The variable DLAYGO is defined equal to the file number.

NOTE: Even if DIC(0) contains L and one of these security checks is passed, LAYGO will not be allowed if a test in the data dictionary's LAYGO node fails.

M If DIC(0) contains M, DIC will do a multiple lookup on all of the file's cross-references from B on to the end of the alphabet. For example, if a given file is cross-referenced both by Name and by Social Security Number, and the user inputs 123-45-6789, DIC, failing to find this input as a Name, will automatically go on to look it up as a Social Security Number.

NOTE: For finer control in specifying the indexes used for lookup, see the alternate lookup entry points IX^DIC and MIX^DIC1.

N If DIC(0) contains N, the input is allowed to be checked as an internal entry number even if the file in question is not normally referenced by number. However, input is only checked as an IEN if no other matches are found during regular lookup.

If DIC(0) does not contain an N, the user is still allowed to select by entry number by preceding the number with the accent grave character (‘). When a **‘** is used, the lookup is limited to internal entry numbers only.

Placing N in DIC(0) does not force IEN interpretation; it only permits it. In order to force IEN interpretation, you must use the accent grave (‘) character.

NOTE: With this flag, when DIC(0) contains an L, users may be allowed to force the internal entry number when adding new entries to the file. If the user enters a number N that is not found on any of the cross-references, and if the .01 field is not numeric and the file is not DINUMed, and if FileMan can talk to the users (DIC(0)["E"), then the user will be asked whether they want to add the new entry, and will be prompted for the value of the .01 field. The entry will be added at the record number N that was originally entered by the user. Note that if there is a .001 field on the file, the number N must also pass the INPUT transform for the .001 field.

O If DIC(0) contains the letter O, then for each index searched, FileMan looks first for exact matches to the lookup value before looking for partial matches. If an exact match is found, then FileMan returns only that match and none of the partial matches on the index. Thus if an index contained the entries 'SMITH,SAM' and 'SMITH,SAMUEL' and if the user typed a lookup value of 'SMITH,SAM', then only the 'SMITH,SAM' entry would be selected, and the user would never see the entry 'SMITH,SAMUEL'. Note that if partial matches but no exact matches are found in the first index(es) searched, but if exact matches are found in an index searched later, then the partial matches from the first index(es) are returned along with the exact match from the later index(es).

Q If DIC(0) contains Q and erroneous input is entered, two question marks (??) will be displayed and a "beep" will sound.

S If DIC(0) does not contain S, the value of the .01 field and Primary Key fields (if the file has a Primary Key) will be displayed for all matches found in any cross-reference. If DIC(0) does contain S, the .01 field and Primary Key fields will not be displayed unless they are one of the indexed fields on which the match was made.

T "T flag in DIC(0). Present every match to the lookup value, quitting only when user either selects one of the presented entries, enters ^^ to quit, or there are no more matching entries found.

Currently, if one or more matches are found in the first pass through the indexes, then FileMan quits the search, whether or not one of the entries is selected. Only if no matches are found in the first pass does FileMan continue on to try transforms to the lookup value. This includes transforms to find internal values of pointers, variable pointers, dates or sets.

Another feature of the "T" flag is that indexes are truly searched in the order requested. If, for example, an index on a pointer field comes before an index on a free-text field, matches from the pointer field will be presented to the user before matches to the free-text field. When used in combination with the "O" flag, all indexes will be searched for an exact match.

Then, only if no matches are found, will FileMan make a second pass through the indexes looking for partial matches.

U	Normally the lookup value is expected to be in external format (for dates, pointers and such). FileMan first searches the requested index for a match to the user input as it was typed in. Then, if no match is found, FileMan automatically tries certain transforms on the lookup value. For instance, if one of the lookup indexes is on a date field, FileMan tries to transform the lookup value to an internal date, then checks the index again. The U flag causes FileMan to look for an exact match on the index and to skip any transforms. Thus the lookup value must be in internal format. This is especially useful for lookups on indexed pointer fields, where the IEN from the pointed-to file is already known. Ordinarily this flag would not be used along with the "A", "B", "M", "N" or "T" flags. In many cases it makes sense to combine this with the "X" flag.
V	If DIC(0) contains V and only one match is made to the user's lookup value, then they will be asked "OK?" and they will have to verify that the looked-up entry is the one they wanted. This is an on the fly way of getting behavior similar to the permanent flag that can be set on a file by answering "YES" to the question "ASK 'OK' WHEN LOOKING UP AN ENTRY?" (See the EDIT FILE option within the FileMan UTILITY option, described in the Advanced User Manual).
X	If DIC(0) contains X, for an exact match, the input value must be found exactly as it was entered. Otherwise, the routine will look for any entries that begin with the input X. Unless 'X-act match' is specified, lowercase input that fails in the lookup will automatically be converted to uppercase, for a second lookup attempt. The difference between X and O (described above) is that X requires an exact match. If there is not one, either DIC exits or tries to add a new entry. With O, if there is not an exact match, DIC looks for a partial match beginning with the input.
Z	If DIC(0) contains Z and if the lookup is successful, then the variable Y(0) will also be returned. It will be set equal to the entire zero node of the entry that has been found. Another array element, Y(0,0), is also returned and will be set equal to

the printable expression of the .01 field of the entry selected. This has no use for Free Text and Numeric data types unless there is an OUTPUT transform. However, for Date/Time, Set of Codes and Pointer data types, Y(0,0) will contain the external format.

Adding New Subentries to a Multiple

You can use ^DIC or FILE^DICN to add new subentries to a multiple. In order to add a subentry, the following variables need to be defined:

DIC	Set to the full global root of the subentry. For example, if the multiple is one level below the top file level: file's_root,entry#,multiple_field's_node,
DIC(0)	Must contain "L" to allow LAYGO.
DIC("P")	Set to the 2nd piece of 0-node of the multiple field's DD entry. **NOTE:** As of Version 22 of FileMan, the developer is no longer required to set DIC("P"). The only exception to this is for a few files that are not structured like a normal FileMan file, where the first subscript of the data is variable in order to allow several different 'globals' to use the same DD. An example of this is the FileMan Audit files where the first subscript is the file number of the file being audited.
DA(1)... **DA(n)**	Set up this array such that DA(1) is the IEN at the next higher file level above the multiple that the lookup is being performed in, DA(2) is the IEN at the next higher file level (if any), ... DA(n) is the IEN at the file's top level. **NOTE:** The value of the unsubscripted DA node should not be defined when doing lookups in a subfile—that's the value you're trying to obtain!

Below is an example of code that:

1. Uses ^DIC to interactively select a top-level record.
2. Uses ^DIC to select or create a **subentry** in a multiple in that record.
3. Uses ^DIE to edit fields in the selected or created subentry.

The file's root in this example is '^DIZ(16150,', the multiple's field number is 9, and the multiple is found on node 4. The code for this example follows:

```
; a call is made to DIC so the user can select an entry in the file
;
S DIC="^DIZ(16150,",DIC(0)="QEAL" D ^DIC
I Y=-1 K DIC Q  ;quit if look-up fails
;
; a second DIC call is set up to select the subentry
;
S DA(1)=+Y ;+Y contains the internal entry number of entry chosen
S DIC=DIC_DA(1)_",4," ;the root of the subfile for that entry
S DIC(0)="QEAL" ;LAYGO to the subfile is allowed
S DIC("P")=$P(^DD(16150,9,0),"^",2) ;returns the subfile# and specifiers
D ^DIC I Y=-1 K DIC,DA Q  ;user selects or adds subentry
;
; a DIE call is made to edit fields in subfile
;
S DIE=DIC K DIC ;DIE now holds the subfile's root
S DA=+Y ;+Y contains the internal entry number of subentry chosen
S DR="1;2" D ^DIE ;edit fields number 1 and 2
K DIE,DR,DA,Y Q
```

IX^DIC: Lookup/Add

This entry point is similar to ^DIC and MIX^DIC1, except for the way it uses cross-references to perform lookup. The three entry points perform lookups as follows:

^DIC	Starts with the B cross-reference, or uses only the B cross-reference [unless K is passed in DIC(0)].
IX^DIC	Starts with the cross-reference you specify or uses only the cross-reference you specify.
MIX^DIC1	Uses the set of cross-references you specify.

Input Variables (Required)

NOTE: All of the input variables described in ^DIC can be used in the IX^DIC call. The following variables are required.

DIC The global root of the file, e.g., ^DIZ(16000.1,.

DIC(0) The lookup parameters as previously described for ^DIC.

D The cross-reference in which to start looking. If DIC(0) contains M, then DIC will continue the search on all other lookup cross-references, in alphabetical order. If it does not, then the lookup is only on the single cross-reference. This variable is killed by VA FileMan; it is undefined when the IX^DIC call is complete.

If DIC(0) contains "L", (i.e., user will be allowed to add a new entry to the file), then either a) D should be set to "B" or b) D should be set to an index that alphabetically comes before "B" and DIC(0) should contain "M" or c) D should contain the name of a compound index.

X If DIC(0) does not contain an A, then the variable X must be defined equal to the value you want to look up.

If the lookup index is compound (i.e., has more than one data subscript), then X can be an array X(n) where "n" represents the position in the subscript. For example, if X(2) is passed in, it will be used as the lookup value to match to the entries in the second subscript of the index. If only the lookup value X is passed, it will be assumed to be the lookup value for the first subscript in the index, X(1).

Input Variables (Optional)

All of the ^DIC input variables can be used in the IX^DIC call. These variables below are optional.

DIC("A"), DIC("B"), DIC("DR"), DIC("P"), DIC("PTRIX",f,p,t)=d DIC("S"), DIC("V"), DIC("W") DIC("?N",file#)=n	This set of input variables affects the behavior of lookup as described for ^DIC.

Output Variables

Y	DIC always returns the variable Y. The variable Y is returned in one of these three formats:	
	Y=-1	The lookup was unsuccessful.
	Y=N^S	N is the Internal Entry Number of the entry in the file and S is the value of the .01 field for that entry.
	Y=N^S^1	N and S are defined as above and the 1 indicates that this entry has just been added to the file.
Y(0)	This variable is only set if DIC(0) contains a Z. When the variable is set, it is equal to the entire zero node of the entry that was selected.	
Y(0,0)	This variable also is only set if DIC(0) contains a Z. When the variable is set, it is equal to the external form of the .01 field of the entry.	

The following are examples of returned Y variables based on a call to the EMPLOYEE file stored in ^EMP(:

```
S DIC="^EMP(",DIC(0)="QEZ",X="SMITH"
D ^DIC
```

Returned are:

```
Y      = "7^SMITH,SAM"
Y(0)   = "SMITH,SAM^M^2231109^2
Y(0,0) = "SMITH,SAM"
```

If the lookup had been done on a file whose .01 field points to the EMPLOYEE file, the returned variables might look like this:

```
Y      = "32^7"  [ Entry #32 in this file and #7 in
                   EMPLOYEE file.]
Y(0)   = "7^RX 2354^ON HOLD"
Y(0,0) = "SMITH,SAM"  [.01 field of entry 7 in
                       EMPLOYEE file]
```

X Contains the value of the field where the match occurred.

If the lookup index is compound (i.e., has more than one data subscript), and if DIC(0) contains an A so that the user is prompted for lookup values, then X will be output as an array X(n) where "n" represents the position in the subscript and will contain the values from the index on which the entry was found. Thus, X(2) would contain the value of the second subscript in the index. If possible, the entries will be output in their external format (i.e., if the subscript is not computed and doesn't have a transform). If the entry is not found on an index (for example, when lookup is done with X=" " [the space-bar return feature]), then X and X(1) will contain the user input, but the rest of the X array will be undefined.

DTOUT This is only defined if DIC has timed-out waiting for input from the user.

DUOUT This is only defined if the user entered an up-arrow.

DO^DIC1: File Info Setup

This entry point retrieves a file's file header node, code to execute its identifiers and its screen (if any), and puts them into local variables for use during lookup into a file.

If $D(DO) is greater than zero, DO^DIC1 will QUIT immediately. If DIC("W") is defined before calling DO^DIC1, it will not be changed.

Input Variables

DIC	The global root of the file, e.g., ^DIZ(16000.1,.
DIC(0)	The lookup parameters as previously described for ^DIC.

Output Variables

DO

File name^file number and specifiers. This is the file header node.

NOTE: Use the letter O, not the number zero, in this variable name.

DO(2)

File number and specifiers. This is the second ^piece of DO. +DO(2) will always equal the file number.

DIC("W")

This is an executable variable which contains the write logic for identifiers. When an entry is displayed, the execution of this variable shows other information to help identify the entry. This variable is created by $ORDERing through the data dictionary ID level, for example:

```
^DD(+DO(2),0,"ID",value)
```

NOTE: The specifier, I, must be in DO(2) for VA FileMan to even look at the ID-nodes.

DO("SCR")

An executable variable which contains a file's screen (if any). The screen is an IF-statement that can screen out certain entries in the file. This differs from DIC("S") in that it is used on every lookup regardless of input or output; that is, the screen is applied to inquiries and printouts as well as to lookups. The value for this variable comes from ^DD(+DO(2),0,"SCR") and the specifier "s" must be in DO(2).

MIX^DIC1: Lookup/Add

This entry point is similar to ^DIC and IX^DIC, except for the way it uses cross-references to do lookup. The three entry points perform lookups as follows:

^DIC	Starts with the B cross-reference or uses only the B cross-reference (unless K is passed in DIC(0)).
IX^DIC	Starts with the cross-reference you specify or uses only the cross-reference you specify.
MIX^DIC1	Uses the set of cross-references you specify.

Input Variables (Required)

NOTE: All of the input variables described in ^DIC can be used in the MIX^DIC1 call. The following variables are required.

DIC The global root of the file, e.g., ^DIZ(16000.1,.

DIC(0) The lookup parameters as previously described for ^DIC.

D The list of cross-references, separated by up-arrows, to be searched, e.g., D="SSN^WARD^B". This variable is killed by VA FileMan; it is undefined when the MIX^DIC1 call is complete. If DIC(0) contains "L", meaning that the user can add a new entry to the file, then either a) the "B" index should be included in the list contained in D, or b) D should be set to the name of a compound index.

Make sure DIC(0) contains M; otherwise, only the first cross-reference in D will be used for the lookup.

X If DIC(0) does not contain an A, then the variable X must be defined equal to the value you want to look up.

If the lookup index is compound (i.e., has more than one data subscript), then X can be an array X(n) where "n" represents the position in the subscript. For example, if X(2) is passed in, it will be used as the lookup value to match to the entries in the second subscript of the index. If only the lookup value X is